МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего

образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

**ОТЧЕТ**

**по Лабораторной работе №4**

по дисциплине

Технология программирования

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Багиров М.Б.

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Вихрова С.С

(подпись) (фамилия, и.,о.)

18-ИСТ-4

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород   
2020

Оглавление

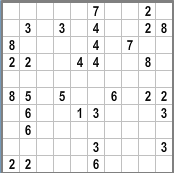
[Вариант №4 3](#_Toc41322033)

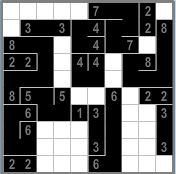
[Листинг программы 4](#_Toc41322034)

[Результат работы программы 12](#_Toc41322035)

# Вариант №4

**Филиппинский кроссворд** (*"Link-a-Pix", "Paint by Pairs"*) - это головоломка с числами. Все числа, расположенные в сетке, кроме единицы, имеют свою пару. Необходимо найти каждую пару чисел и соединить их линиями. Количество клеток в ней должно равняться числам на ее концах. Линии, соединяющие пары, могут преломляться и идти в горизонтальном или вертикальном направлениях (но не по диагоналям). Линии не могут пересекаться друг с другом или проходить через одни и те же клетки.





# Листинг программы

-- Game.cs–

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace GameOver

{

public partial class Game : Form

{

List<DataGridViewCell> AllCells = new List<DataGridViewCell>(); // все ячейки с числами на данном поле

int countAlgoritm = 0; // количество запусков алгоритма заполнения поля

List<Tree> listCells = new List<Tree>();

public Game()

{

InitializeComponent();

Colums();//вызываем функцию, которая создаёт поле

ReadFile();//вызываем функцию, которая читает данные из файла

CountAllCellls();// считываем все ячейки с числами

}

//функция для чтения из файла

public void ReadFile()

{

Random random = new Random();

int x = random.Next(1, 4);

string nameFile = “Game” + 5 + “.txt”;

FileStream file = new FileStream(nameFile, FileMode.Open, FileAccess.Read);

StreamReader reader = new StreamReader(file);

int k = -1; // переменная для определения строки в поле

string text;

while ((text = reader.ReadLine()) != null)

{ //чтение по строчно

k = k + 1;

string[] splitLine = text.Split(‘ ‘); // разделитель

for (int I = 0; I < splitLine.Length; i++)

{

if (splitLine[i] != “0”)

{

dataGridView1.Rows[k].Cells[i].Value = splitLine[i];

dataGridView1.Rows[k].Cells[i].Style.BackColor = Color.White;

}

if (splitLine[i] == “1”)

{

dataGridView1.Rows[k].Cells[i].Style.BackColor = Color.Black;

dataGridView1.Rows[k].Cells[i].Style.ForeColor = Color.White;

}

}

}

reader.Close();

file.Close();

}

//функция для разбиения поля на ячейки

public void Colums()

{

int k = this.dataGridView1.Size.Width / 30;

System.Windows.Forms.DataGridViewColumn[] f = new System.Windows.Forms.DataGridViewColumn[] { };

for (int I = 0; I < k; i++)

{

System.Windows.Forms.DataGridViewTextBoxColumn co = new System.Windows.Forms.DataGridViewTextBoxColumn();

co.Width = 30;

this.dataGridView1.Columns.Add(co);

}

k = this.dataGridView1.Size.Height / 30;

for (int I = 0; I < k – 1; i++)

{

this.dataGridView1.Rows.Add();

}

}

//фукнцтя для прохождения по всему полю и запоминание всех ячеек с числами

public void CountAllCellls()

{

Console.WriteLine(«Количсетво строк « + this.dataGridView1.Rows.Count + «Количсетво столбцов « + this.dataGridView1.Columns.Count);

for (int c0 = 0; c0 < this.dataGridView1.Rows.Count; c0++)

{

for (int c1 = 0; c1 < this.dataGridView1.Columns.Count; c1++)

{

if (dataGridView1[c1, c0].Value != null && dataGridView1[c1, c0].Value.ToString() != “” && dataGridView1[c1, c0].Value.ToString() != “1”) {

Console.WriteLine(“Индексы “ + c0 +”, “+ c1);

AllCells.Add(dataGridView1[c1, c0]); }

}

}

}

//основной алгоритм нахождения пар

public void Algoritm()

{

Console.WriteLine(«Количество ячеек без пары « + AllCells.Count());

for (int c0 = 0; c0 < this.dataGridView1.Rows.Count; c0++)

{

for (int c1 = 0; c1 < this.dataGridView1.Columns.Count; c1++)

{

//проверка на не пустую ячейку

if (this.dataGridView1[c1, c0].Value != null && this.dataGridView1.Rows[c0].Cells[c1].Value.ToString() != “” && Convert.ToInt32(this.dataGridView1.Rows[c0].Cells[c1].Value) >= 1 && this.dataGridView1.Rows[c0].Cells[c1].Style.BackColor != Color.Black)

{

Console.WriteLine(“Какая ячейка проверяется: “ + c0 + “, “ + c1);

List<DataGridViewCell> nn = new List<DataGridViewCell>();//массив совпавших ячеек

//проход по всем соседним ячейкам к выбранной ячейки в шахматном порядке

//Например, к ячейке в середине будут проверятся все ячейки с плюсами

// |\_|\_|+|\_|\_|

// |\_|+|\_|+|\_|

// |+|\_|3|\_|+|

// |\_|+|\_|+|\_|

// |\_|\_|+|\_|\_|

for (int i = c0 – (Convert.ToInt32(this.dataGridView1.Rows[c0].Cells[c1].Value) – 1); i < c0 + Convert.ToInt32(this.dataGridView1.Rows[c0].Cells[c1].Value); i++)

{

for (int j = c1 – ((Convert.ToInt32(this.dataGridView1.Rows[c0].Cells[c1].Value) – 1) – Math.Abs(I – c0)); j <= c1 + ((Convert.ToInt32(this.dataGridView1.Rows[c0].Cells[c1].Value) – 1) – Math.Abs(I – c0)); j = j + 2)

{

if (I >= 0 && j >= 0 && I < this.dataGridView1.Rows.Count && j < this.dataGridView1.ColumnCount)

if (this.dataGridView1[j, i].Value != null && this.dataGridView1[j, i] != this.dataGridView1[c1, c0] && this.dataGridView1[j, i].Value.ToString() == this.dataGridView1.Rows[c0].Cells[c1].Value.ToString() && this.dataGridView1.Rows[i].Cells[j].Style.BackColor == Color.White)

{

nn.Add(this.dataGridView1[j, i]);

}

}

}

//если можно сразу соединить эти две ячейки

//по вертикали

if (nn.Count > 0 )

{

//Создание объекта класса Tree

Tree parent = new Tree(dataGridView1, this.dataGridView1.Rows[c0].Cells[c1], Convert.ToInt32(this.dataGridView1.Rows[c0].Cells[c1].Value), this.dataGridView1.Rows[c0].Cells[c1].Value.ToString());

parent.CalculatingMoves(this.dataGridView1.Rows[c0].Cells[c1]);//функция вычисления всех возможны[ ходов

Console.WriteLine(« ======== Дети =========»);

int countLastNode = 1;//количество всех путей, которые могут образовать пару

//parent.LastNode(ref countLastNode);

if (countLastNode == 1)

{

List<DataGridViewCell> \_moves = new List<DataGridViewCell>();

List<DataGridViewCell> \_moves2 = new List<DataGridViewCell>();

parent.AllChild(\_moves, \_moves2, ref parent.shelders);

parent.listSelectedPath = \_moves2;

parent.countMatchingCells = nn.Count();

if (\_moves2.Count() > 1)

{

AllCells.Remove(dataGridView1[c1, c0]);

AllCells.Remove(dataGridView1[\_moves2[\_moves2.Count() – 1].ColumnIndex, \_moves2[\_moves2.Count() – 1].RowIndex]);

}

for (int I = 0; I < \_moves2.Count(); i++)

{

dataGridView1[\_moves2[i].ColumnIndex, \_moves2[i].RowIndex].Style.BackColor = Color.Black;

dataGridView1[\_moves2[i].ColumnIndex, \_moves2[i].RowIndex].Style.ForeColor = Color.White;

}

}

listCells.Add(parent);

}

}

}

}

Console.WriteLine(“////Количество ячеек без пары “ + AllCells.Count() + “////”);

}//Функция для нахождения ячеек, по значению совпавших с данной.

//Функция вернёт все найденный ячейки

//Фукнция находит все ячейки, которые не были раньше выбраны для составления пары

public List<DataGridViewCell> ChildWhithWhite( DataGridViewCell data) {

List<DataGridViewCell> nn = new List<DataGridViewCell>();//массив совпавших ячеек

for (int I = data.RowIndex – (Convert.ToInt32(this.dataGridView1.Rows[data.RowIndex].Cells[data.ColumnIndex].Value) – 1); I < data.RowIndex + Convert.ToInt32(this.dataGridView1.Rows[data.RowIndex].Cells[data.ColumnIndex].Value); i++)

{

for (int j = data.ColumnIndex – ((Convert.ToInt32(this.dataGridView1.Rows[data.RowIndex].Cells[data.ColumnIndex].Value) – 1) – Math.Abs(I – data.RowIndex)); j <= data.ColumnIndex + ((Convert.ToInt32(this.dataGridView1.Rows[data.RowIndex].Cells[data.ColumnIndex].Value) – 1) – Math.Abs(I – data.RowIndex)); j = j + 2)

{

if (I >= 0 && j >= 0 && I < this.dataGridView1.Rows.Count && j < this.dataGridView1.ColumnCount)

if (this.dataGridView1[j, i].Value != null && this.dataGridView1[j, i] != this.dataGridView1[data.ColumnIndex, data.RowIndex] && this.dataGridView1[j, i].Value.ToString() == this.dataGridView1.Rows[data.RowIndex].Cells[data.ColumnIndex].Value.ToString() && this.dataGridView1[j, i].Style.BackColor != Color.Black)

{

nn.Add(this.dataGridView1[j, i]);

}

}

}

return nn;

}

//Функция для нахождения ячеек, по значению совпавших с данной.

//Функция вернёт все найденный ячейки

//Фукнция находит все ячейки, которые были раньше выбраны для составления пары и у которых BackColor равен Black

public List<DataGridViewCell> ChildWhithBlack(DataGridViewCell data)

{

List<DataGridViewCell> nn = new List<DataGridViewCell>();//массив совпавших ячеек

for (int I = data.RowIndex – (Convert.ToInt32(this.dataGridView1.Rows[data.RowIndex].Cells[data.ColumnIndex].Value) – 1); I < data.RowIndex + Convert.ToInt32(this.dataGridView1.Rows[data.RowIndex].Cells[data.ColumnIndex].Value); i++)

{

for (int j = data.ColumnIndex – ((Convert.ToInt32(this.dataGridView1.Rows[data.RowIndex].Cells[data.ColumnIndex].Value) – 1) – Math.Abs(I – data.RowIndex)); j <= data.ColumnIndex + ((Convert.ToInt32(this.dataGridView1.Rows[data.RowIndex].Cells[data.ColumnIndex].Value) – 1) – Math.Abs(I – data.RowIndex)); j = j + 2)

{

if (I >= 0 && j >= 0 && I < this.dataGridView1.Rows.Count && j < this.dataGridView1.ColumnCount)

if (this.dataGridView1[j, i].Value != null && this.dataGridView1[j, i] != this.dataGridView1[data.ColumnIndex, data.RowIndex] && this.dataGridView1[j, i].Value.ToString() == this.dataGridView1.Rows[data.RowIndex].Cells[data.ColumnIndex].Value.ToString() && this.dataGridView1[j, i].Style.BackColor == Color.Black)

{

nn.Add(this.dataGridView1[j, i]);

}

}

}

return nn;

}//Алгоритм для редактирования полученных пар, если остались ячейки без пары

public void Algoritm2()

{

Console.WriteLine(“///////////// Algoritm2 //////////////”);

//Лист AllCells представляет из себя массив всех ячеек без пары

//В процессе Algoritm() данный массив будет уменьшаться и останутся только ячеек без пары

for (int i=0; I < AllCells.Count(); i++)

{

Tree cell = new Tree(dataGridView1, dataGridView1[AllCells[i].ColumnIndex, AllCells[i].RowIndex], Convert.ToInt32(dataGridView1[AllCells[i].ColumnIndex, AllCells[i].RowIndex].Value.ToString()), dataGridView1[AllCells[i].ColumnIndex, AllCells[i].RowIndex].Value.ToString());

List<DataGridViewCell> childCells = new List<DataGridViewCell>();

int flag = 1;//флаг для определения, для данной ячейки были выбраны ячейки, которые уже являются чей то парой или нет

childCells = ChildWhithWhite(cell.thisNodeCell);//возвращение массива с совпавшими ячейками с белым фоном

//проверка на то, нашлись ли такие ячейки

//если их количество равно 0, то будут проверяться ячейки, которые уже образуют с кем то пару

if (childCells.Count() == 0) {

childCells = ChildWhithBlack(cell.thisNodeCell);

flag = -1;

}

//функция для нахождения всех возможных путей

cell.CalculatingMoves2(cell.thisNodeCell, flag);

//вспомогательные листы для нахождения всех конечных путей, которые в итоге могут образовать пару

List<DataGridViewCell> \_moves = new List<DataGridViewCell>();

List<DataGridViewCell> \_moves2 = new List<DataGridViewCell>();

//нахождения этих конечных путей

cell.AllChild(\_moves, \_moves2, ref cell.shelders);

//перебор всех полученных конечных путей и определение того пути, который приведёт к нужной ячейке

foreach (List<DataGridViewCell> data in cell.shelders) {

if (data[data.Count() – 1] == childCells[0]) {

cell.listSelectedPath = data;

}

}

//прохождение по получившемуся листу с возможным путём

for (int count = 0; count < cell.listSelectedPath.Count(); count++) {

//если при прохождении пути на поле в данном месте уже черная ячейка

if (dataGridView1[cell.listSelectedPath[count].ColumnIndex, cell.listSelectedPath[count].RowIndex].Style.BackColor == Color.Black)

{

//то проходит по массиву всех путей уже имеющихся путей и находим тот, к кторому относится эта ячейка

for (int j = 0; j < listCells.Count(); j++)

{

for (int h = 0; h < listCells[j].listSelectedPath.Count(); h++)

{

if (cell.listSelectedPath[count] == listCells[j].listSelectedPath[h] ) {

int countMoves = 0;

listCells[j].LastNode(ref countMoves);

if (countMoves > 0) {

//убираем данный путь на поле

foreach (DataGridViewCell data in listCells[j].listSelectedPath) {

this.dataGridView1[data.ColumnIndex, data.RowIndex].Style.BackColor = Color.White;

this.dataGridView1[data.ColumnIndex, data.RowIndex].Style.ForeColor = Color.Black;

}

//если ячейка, с которой проверяется ячейка на поле является либо началбно, либо последней (т.е. она идёт со значением)

//то указываем у конфликтного пути, что его новый путь будет один из возможных

if (h != 0 && h != listCells[j].listSelectedPath.Count() – 1)

{

foreach (List<DataGridViewCell> list in listCells[j].shelders)

{

if (list != listCells[j].listSelectedPath)

{

listCells[j].listSelectedPath = list;

}

}

for (int g = 0; g < listCells[j].listSelectedPath.Count(); g++)

{

dataGridView1[listCells[j].listSelectedPath[g].ColumnIndex, listCells[j].listSelectedPath[g].RowIndex].Style.BackColor = Color.Black;

dataGridView1[listCells[j].listSelectedPath[g].ColumnIndex, listCells[j].listSelectedPath[g].RowIndex].Style.ForeColor = Color.White;

}

}

//если же проверяемые ячейки всё-таки совпали с начальными, то добавляем начальные и конечные ячейки из конфликтного пути в массив ячеек без пары

if (h == 0) AllCells.Add(listCells[j].listSelectedPath[listCells[j].listSelectedPath.Count() – 1]);

if (h == listCells[j].listSelectedPath.Count() – 1) AllCells.Add(listCells[j].listSelectedPath[0]);

//и закрашиваем выбранный путь для изначальной ячейки, к которой искали пару

for (int g = 0; g < cell.listSelectedPath.Count(); g++)

{

dataGridView1[cell.listSelectedPath[g].ColumnIndex, cell.listSelectedPath[g].RowIndex].Style.BackColor = Color.Black;

dataGridView1[cell.listSelectedPath[g].ColumnIndex, cell.listSelectedPath[g].RowIndex].Style.ForeColor = Color.White;

}

AllCells.Remove(cell.thisNodeCell);

AllCells.Remove(cell.listSelectedPath[cell.listSelectedPath.Count() – 1]);

}

}

}

}

}

}

listCells.Add(cell);

}

Console.WriteLine(AllCells.Count());

}private void ButtonStart\_Click(object sender, EventArgs e)

{

while (countAlgoritm < 4 || AllCells.Count() !=0 )

{

//пока количество ячеек с числоми не равно 0 и количество запусков не превышает 5

Algoritm();//запускаем алгоритм нахождения пар

MessageBox.Show(«Game»);

/\*if (AllCells.Count() == 1)

{

MessageBox.Show(«Не возможно пройти!»);

}

else\*/

//елси после 5 попыток остались не задйствованные ячейки, то выводим сооющение

if (AllCells.Count() != 0 && countAlgoritm < 4)

{

Algoritm2();

}

++countAlgoritm;

//if (countAlgoritm >= 3 && AllCells.Count() == 0) MessageBox.Show(“Игра закончена!”);

//else if (countAlgoritm >= 3 && AllCells.Count > 0) MessageBox.Show(“Не возможно пройти!”);

MessageBox.Show(«Game»);

}

}

}

}

--Tree.cs--

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace GameOver

{

public class Tree : Form

{

DataGridView dataGridView1;

public DataGridViewCell thisNodeCell;

List<Tree> movesCell = new List<Tree>();// все возможные ходы

public List<DataGridViewCell> listSelectedPath = new List<DataGridViewCell>();

public List<List<DataGridViewCell>> shelders = new List<List<DataGridViewCell>>();

int count;//переменная отвечающая за то, сколько ещё ячеек нужно до составления пары

string value;//значение выбранной ячейки

public int countMatchingCells = 0;

int IndexRowMain;

int IndexColMain;

public Tree(DataGridView dv, DataGridViewCell cell, int \_count, string \_value, int Row, int Col)

{

dataGridView1 = dv;

thisNodeCell = cell;

count = \_count;

value = \_value;

IndexColMain = Col;

IndexRowMain = Row;

listSelectedPath = new List<DataGridViewCell>();

}

public Tree(DataGridView dv, DataGridViewCell cell, int \_count, string \_value)

{

dataGridView1 = dv;

thisNodeCell = cell;

count = \_count;

value = \_value;

IndexColMain = cell.ColumnIndex;

IndexRowMain = cell.RowIndex;

listSelectedPath = new List<DataGridViewCell>();

}

//функция для отыскания единственного возможного пути для соединения ячеек

public void AllChild(List<DataGridViewCell> moves, List<DataGridViewCell> \_Moves, ref List<List<DataGridViewCell>> list)

{

if (movesCell.Count() != 0 )

{

moves.Add(thisNodeCell);

if (count == 2 && \_Moves.Count() == 0)

{

\_Moves.Clear();

Console.WriteLine("Приравнинвание ");

for (int i = 0; i < moves.Count; i++) {

\_Moves.Add(moves[i]);

}

\_Moves.Add(movesCell[0].thisNodeCell);

list.Add(\_Moves);

}

if (count == 2 && \_Moves.Count() > 0) {

List<DataGridViewCell> Moves\_ = new List<DataGridViewCell>();

for (int i = 0; i < moves.Count; i++)

{

Moves\_.Add(moves[i]);

}

Moves\_.Add(movesCell[0].thisNodeCell);

list.Add(Moves\_);

}

/\*if (count == 2 && \_Moves.Count() > 0)

{

\_Moves.RemoveRange(0, \_Moves.Count() - 1);

for (int i = 0; i < moves.Count; i++) \_Moves.Add(moves[i]);

\_Moves.Add(movesCell[0].thisNodeCell);

}\*/

for (int i = 0; i < movesCell.Count(); i++)

{

movesCell[i].AllChild(moves, \_Moves, ref list);

}

moves.Remove(thisNodeCell);

}

}//функция для проверки на то, что путь соединения ячеек один

public void LastNode(ref int countLast)

{

if (movesCell.Count() != 0)

{

if (count == 2 && movesCell.Count() == 1)

{

++countLast;

Console.WriteLine("Добавление" + countLast);

}

Console.WriteLine("Раскрытие детей");

for (int i = 0; i < movesCell.Count(); i++)

{

movesCell[i].LastNode(ref countLast);

}

}

}

//функция для вычисления всех путей, которые можно построить от выбранной ячейки

public void CalculatingMoves(DataGridViewCell parent)

{

if (count > 2)

{

for (int i = -1; i < 2; i++)

{

if (i != 0)

{

if (thisNodeCell.RowIndex + i >= 0 && thisNodeCell.RowIndex + i < this.dataGridView1.RowCount)

{

//если ячейка пустая и у нее белый цвет фона

if (this.dataGridView1.Rows[thisNodeCell.RowIndex + i].Cells[thisNodeCell.ColumnIndex].Value == null && this.dataGridView1[thisNodeCell.ColumnIndex, thisNodeCell.RowIndex + i].Style.BackColor != Color.Black)

{

Console.WriteLine("gfgfgfgfgfgfgfgfg " + (thisNodeCell.RowIndex + i) + ", " + thisNodeCell.ColumnIndex);

Console.WriteLine("gfgfgfgfgfgfgfgfg " + parent.RowIndex + ", " + parent.ColumnIndex);

if (thisNodeCell.RowIndex + i != parent.RowIndex)

{

int c = count - 1;

Tree child = new Tree(dataGridView1, this.dataGridView1.Rows[thisNodeCell.RowIndex + i].Cells[thisNodeCell.ColumnIndex], c, value, IndexRowMain, IndexColMain);

child.CalculatingMoves(thisNodeCell);

movesCell.Add(child);

}

}

}

if (thisNodeCell.ColumnIndex + i >= 0 && thisNodeCell.ColumnIndex + i < this.dataGridView1.ColumnCount)

{

Console.WriteLine("Count moves " + movesCell.Count());

//если ячейкая пустая и белого цвета

if (this.dataGridView1[thisNodeCell.ColumnIndex + i, thisNodeCell.RowIndex].Value == null && dataGridView1[thisNodeCell.ColumnIndex + i, thisNodeCell.RowIndex].Style.BackColor != Color.Black)

{

Console.WriteLine("gfgfgfgfgfgfgfgfg " + (thisNodeCell.RowIndex) + ", " + (thisNodeCell.ColumnIndex + i));

Console.WriteLine("gfgfgfgfgfgfgfgfg " + parent.RowIndex + ", " + parent.ColumnIndex);

if (thisNodeCell.ColumnIndex + i != parent.ColumnIndex)

{

int c = count - 1;

Tree child = new Tree(dataGridView1, this.dataGridView1.Rows[thisNodeCell.RowIndex].Cells[thisNodeCell.ColumnIndex + i], c, value, IndexRowMain, IndexColMain);

child.CalculatingMoves(thisNodeCell);

movesCell.Add(child);

}

}

}

}

}

}

//если это предпоследний ход

//то проверяется, есть ли вокруг свободная ячейка с нужным значение

if (count == 2)

{

for (int j = -1; j < 2; j++)

{

if (j != 0 && thisNodeCell.RowIndex + j >= 0 && thisNodeCell.RowIndex + j < this.dataGridView1.RowCount)

{

Console.WriteLine("----------" + count + ", " + value);

Console.WriteLine("То что проверяется" + (thisNodeCell.RowIndex + j) + ", " + thisNodeCell.ColumnIndex);

if (dataGridView1[thisNodeCell.ColumnIndex, thisNodeCell.RowIndex + j].Value != null)

{

if ((thisNodeCell.ColumnIndex != IndexColMain || thisNodeCell.RowIndex + j != IndexRowMain) && dataGridView1[thisNodeCell.ColumnIndex, thisNodeCell.RowIndex + j].Style.BackColor != Color.Black && dataGridView1[thisNodeCell.ColumnIndex, thisNodeCell.RowIndex + j].Value.ToString() == value)

{

Console.WriteLine("Последний ход добавление");

Console.WriteLine("Индексы ячейки от которой проверяется " + thisNodeCell.RowIndex + ", " + thisNodeCell.ColumnIndex);

Tree child = new Tree(dataGridView1, dataGridView1[thisNodeCell.ColumnIndex, thisNodeCell.RowIndex + j], 1, value);

movesCell.Add(child);

}

}

}

if (j != 0 && thisNodeCell.ColumnIndex + j >= 0 && thisNodeCell.ColumnIndex + j < this.dataGridView1.RowCount)

{

Console.WriteLine("----------" + count + ", " + value);

Console.WriteLine("То что проверяется " + (thisNodeCell.RowIndex) + ", " + (thisNodeCell.ColumnIndex + j));

if (dataGridView1[thisNodeCell.ColumnIndex + j, thisNodeCell.RowIndex].Value != null)

if ((thisNodeCell.ColumnIndex + j != IndexColMain || thisNodeCell.RowIndex != IndexRowMain) && dataGridView1[thisNodeCell.ColumnIndex + j, thisNodeCell.RowIndex].Style.BackColor != Color.Black && dataGridView1[thisNodeCell.ColumnIndex + j, thisNodeCell.RowIndex].Value.ToString() == value)

{

Console.WriteLine("Последний ход добавление");

Console.WriteLine("Индексы ячейки от которой проверяется " + thisNodeCell.RowIndex + ", " + thisNodeCell.ColumnIndex);

Tree child = new Tree(dataGridView1, dataGridView1[thisNodeCell.ColumnIndex + j, thisNodeCell.RowIndex], 1, value);

movesCell.Add(child);

}

}

}

}

}

public void CalculatingMoves2(DataGridViewCell parent, int flag)

{

if (count > 2)

{

for (int i = -1; i < 2; i++)

{

if (i != 0)

{

if (thisNodeCell.RowIndex + i >= 0 && thisNodeCell.RowIndex + i < this.dataGridView1.RowCount)

{

//если ячейка пустая и у нее белый цвет фона

if (this.dataGridView1.Rows[thisNodeCell.RowIndex + i].Cells[thisNodeCell.ColumnIndex].Value == null )

{

Console.WriteLine("gfgfgfgfgfgfgfgfg " + (thisNodeCell.RowIndex + i) + ", " + thisNodeCell.ColumnIndex);

Console.WriteLine("gfgfgfgfgfgfgfgfg " + parent.RowIndex + ", " + parent.ColumnIndex);

if (thisNodeCell.RowIndex + i != parent.RowIndex)

{

int c = count - 1;

Tree child = new Tree(dataGridView1, this.dataGridView1.Rows[thisNodeCell.RowIndex + i].Cells[thisNodeCell.ColumnIndex], c, value, IndexRowMain, IndexColMain);

child.CalculatingMoves2(thisNodeCell, flag);

movesCell.Add(child);

}

}

}

if (thisNodeCell.ColumnIndex + i >= 0 && thisNodeCell.ColumnIndex + i < this.dataGridView1.ColumnCount)

{

Console.WriteLine("Count moves " + movesCell.Count());

//если ячейкая пустая и белого цвета

if (this.dataGridView1[thisNodeCell.ColumnIndex + i, thisNodeCell.RowIndex].Value == null )

{

Console.WriteLine("gfgfgfgfgfgfgfgfg " + (thisNodeCell.RowIndex) + ", " + (thisNodeCell.ColumnIndex + i));

Console.WriteLine("gfgfgfgfgfgfgfgfg " + parent.RowIndex + ", " + parent.ColumnIndex);

if (thisNodeCell.ColumnIndex + i != parent.ColumnIndex)

{

int c = count - 1;

Tree child = new Tree(dataGridView1, this.dataGridView1.Rows[thisNodeCell.RowIndex].Cells[thisNodeCell.ColumnIndex + i], c, value, IndexRowMain, IndexColMain);

child.CalculatingMoves2(thisNodeCell, flag);

movesCell.Add(child);

}

}

}

}

}

}

//если это предпоследний ход

//то проверяется, есть ли вокруг свободная ячейка с нужным значение

if (count == 2)

{

for (int j = -1; j < 2; j++)

{

if (j != 0 && thisNodeCell.RowIndex + j >= 0 && thisNodeCell.RowIndex + j < this.dataGridView1.RowCount)

{

Console.WriteLine("----------" + count + ", " + value);

Console.WriteLine("То что проверяется" + (thisNodeCell.RowIndex + j) + ", " + thisNodeCell.ColumnIndex);

if (dataGridView1[thisNodeCell.ColumnIndex, thisNodeCell.RowIndex + j].Value != null)

{

Color color;

if (flag == 1) color = Color.White;

else color = Color.Black;

if ((thisNodeCell.ColumnIndex != IndexColMain || thisNodeCell.RowIndex + j != IndexRowMain) && dataGridView1[thisNodeCell.ColumnIndex, thisNodeCell.RowIndex + j].Value.ToString() == value && dataGridView1[thisNodeCell.ColumnIndex, thisNodeCell.RowIndex + j].Style.BackColor == color)

{

Console.WriteLine("Последний ход добавление");

Console.WriteLine("Индексы ячейки от которой проверяется " + thisNodeCell.RowIndex + ", " + thisNodeCell.ColumnIndex);

Tree child = new Tree(dataGridView1, dataGridView1[thisNodeCell.ColumnIndex, thisNodeCell.RowIndex + j], 1, value);

movesCell.Add(child);

}

}

}

if (j != 0 && thisNodeCell.ColumnIndex + j >= 0 && thisNodeCell.ColumnIndex + j < this.dataGridView1.RowCount)

{

Console.WriteLine("----------" + count + ", " + value);

Console.WriteLine("То что проверяется " + (thisNodeCell.RowIndex) + ", " + (thisNodeCell.ColumnIndex + j));

if (dataGridView1[thisNodeCell.ColumnIndex + j, thisNodeCell.RowIndex].Value != null)

{

Color color;

if (flag == 1) color = Color.White;

else color = Color.Black;

if ((thisNodeCell.ColumnIndex + j != IndexColMain || thisNodeCell.RowIndex != IndexRowMain) && dataGridView1[thisNodeCell.ColumnIndex + j, thisNodeCell.RowIndex].Value.ToString() == value && dataGridView1[thisNodeCell.ColumnIndex + j, thisNodeCell.RowIndex].Style.BackColor == color)

{

Console.WriteLine("Последний ход добавление");

Console.WriteLine("Индексы ячейки от которой проверяется " + thisNodeCell.RowIndex + ", " + thisNodeCell.ColumnIndex);

Tree child = new Tree(dataGridView1, dataGridView1[thisNodeCell.ColumnIndex + j, thisNodeCell.RowIndex], 1, value);

movesCell.Add(child);

}

}

}

}

}

}

}

}

# Результат работы программы

